A Numerical Algorithm for the Solution of Viscous Incompressible Flow on GPU's

Santiago Costarelli[1], Mario Storti[1], Rodrigo Paz[1], Lisandro Dalcín[1], and Sergio Idelsohn[1,2,3]

[1] CIMEC-INTEC-CONICET-UNL, Guemes 3450, 3000 Santa Fe, Argentina
santi.costarelli@gmail.com, http://www.cimec.org.ar
[2] International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia (UPC), Gran Capitán s/n, 08034 Barcelona, Spain,
[3] Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain

Abstract. Graphic Processing Units have received much attention in last years. Compute-intensive algorithms operating on multidimensional arrays that have nearest neighbor dependency and/or exploit data locality can achieve massive speedups. This work discuss a solver for the pressure problem in applications using immersed boundary techniques in order to account for moving solid bodies. The solver is based on standard Conjugate Gradients iterations and depends on the availability of a fast Poisson solver on the whole domain to define a preconditioner.

Keywords: Graphics Processing Units; Incompressible Navier-Stokes; Poisson equation

1 Introduction

Graphics Processing Units (GPU) are computer co-processors used in desktop computers and workstations to off-load the renderization of complex graphics from the main processor (CPU). They have evolved to complex systems containing many processing units, a large amount of on-board memory and a computing power in the order of teraflops. They are instances of massively parallel architectures and Single Instruction Multiple Data (SIMD) paradigms.

Recently, GPU's are becoming increasingly popular among scientists and engineers for High Performance Computing (HPC) applications [1–3, 8, 9, 11–15, 19, 20]. This tendency motivated GPU manufacturers to develop General Purpose Graphics Processing Units (GPGPU) targeting the HPC market.

In the pursuit of more realistic visualization algorithms for video games and special effects, solving Partial Differential Equations (PDE) has become a necessary ingredient [6, 7, 10, 18, 22]. Numerical schemes employed in these applications usually sacrifice accuracy for speed, resulting in very fast implementations when comparing to engineering codes.

The resolution of Computational Fluid Dynamics (CFD) problems on GPU's requires of specialized algorithms due to the particular hardware architecture of these devices. Algorithms that fall in the category of Cellular Automata (CA) are the best fitted for GPU's. For instance, explicit Finite Volume or Finite Element methods, jointly with *immersed boundary* techniques [21] to represent solid bodies, can be used on structured cartesian meshes. In the case of incompressible flows, it is not possible to develop a purely explicit algorithm, due to the essentially non-local nature of the incompressibility condition.

Segregated algorithms solve an implicit Poisson equation for the pressure field, being this stage the most time-consuming in the solution procedure. Using fast Poisson solvers like Multigrid (MG) or Fast Fourier Transform (FFT) is tempting but treating moving solid bodies becomes cumbersome in the case of MG or unsuitable for FFT. To surpass these difficulties, Molemaker *et.al* proposed in [13] the Iterated Orthogonal Projection (IOP) method which requires a series of projections on the complete grid (fluid and solid) to enforce the incompressibility and boundary conditions.

In this work an alternative to IOP, the Accelerated Global Preconditioning (AGP), is proposed. The solver is based on using a Preconditioned Conjugate Gradients (PCG) algorithm, so that, it is an accelerated iterative method in contrast to the stationary scheme used in IOP. In addition, AGP method iterates only on pressure, whereas IOP iterates on both pressure and velocity.

2 The Accelerated Global Preconditioning

A preconditioning for embedded problems that is based on solving the problem in the complete mesh is presented. Suppose a situation like in figure 1, with a solid body described by the boundary $\Gamma_{\rm bdy}$. This is embedded in a structured FEM grid of constant mesh size h. The Poisson problem *outside* the body has to be solved, so that this is done by assembling the matrices of those finite elements that are in the fluid region. In order to do that, the center of the elements are checked wether they fall inside or outside the body. In this way the body is approximated by a *staircase geometry* as is shown in gray in the figure. In a FEM context the imposition of the homogeneous Neumann condition is done by simply assembling only those elements that are in the fluid part (filled in gray in the figure). The other elements that are not in gray are *ghost elements* and are not assembled for the solution of the Poisson problem. Only the pressure in the nodes connected to some element that is assembled are relevant, i.e. those that are marked in blue and red. Those that are marked in green are *ghost* and then they are not computed. Those that are computed are classified as *interior* and *boundary*. interior to the fluid, (subindex F) are those that are surrounded by computed elements, or conversely that are not connected to ghost elements. The rest are marked as *boundary* (subindex B, filled in red in the figure). So the Poisson problem is

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

and the splitting of nodes induces a matricial splitting like this

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \mathbf{A}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{x}_F \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_F \\ \mathbf{b}_B \end{bmatrix} \tag{2}$$

For the definition of the preconditioning P, the whole matrix $\tilde{P}$ for all elements (fluid and ghost) is assembled, and the right hand side vector is extended as 0 on the

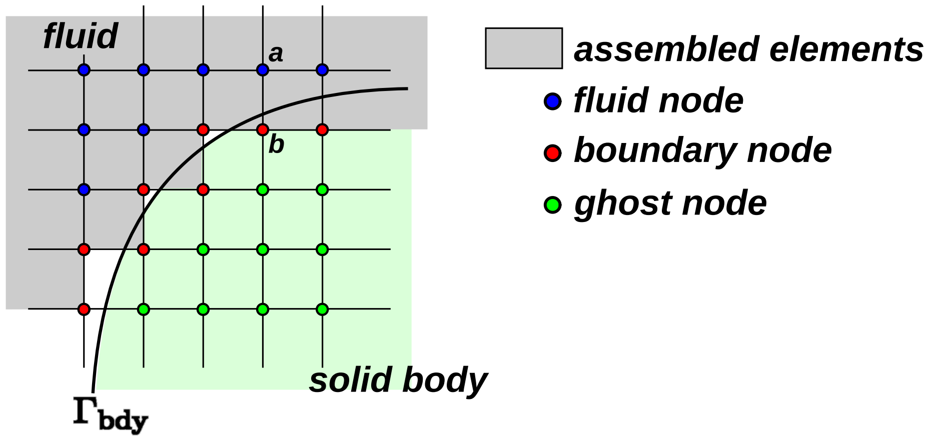

Fig. 1. Description of nodes and elements used in the AGP preconditioning.

ghost elements. The system is solved and then the ghost values are discardeed, i.e. the preconditioning is defined formally as $\mathbf{y}_{FB} = \mathbf{P}\mathbf{x}_{FB}$, where $\mathbf{y}_{FB}$ is the solution of

$$\begin{bmatrix} \tilde{\mathbf{P}}_{FF} & \tilde{\mathbf{P}}_{FB} & \tilde{\mathbf{P}}_{FG} \\ \tilde{\mathbf{P}}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \tilde{\mathbf{P}}_{GF} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_{G} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_{G} \end{bmatrix}. \tag{3}$$

Note that a differente symbol $\tilde{\mathbf{P}}$ is used for the discrete Laplace operator in this equation, since it is assembled on differente elements. However it can be seen that

- $\tilde{\mathbf{P}}_{FF} = \mathbf{A}_{FF}$ since the F nodes are those for which all elements are assembled in the Poisson problem.
- $\tilde{\mathbf{P}}_{FB} = \mathbf{A}_{FB}$, and $\tilde{\mathbf{P}}_{BF} = \mathbf{A}_{BF}$ since for instance, such a coefficient would link nodes as a and b in the figure. This coefficient comes from the assembly of all the elements that are connected to a *and* b, but since a is an F node, it means that all elements connected to a are assembled.
- $\tilde{\mathbf{P}}_{FG} = \tilde{\mathbf{P}}_{GF} = 0$ since F nodes are only connected to fluid elements and G are only connected to ghost elements, so that they can not share an element.

So

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} & \mathbf{0} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} & \tilde{\mathbf{P}}_{BG} \\ \mathbf{0} & \tilde{\mathbf{P}}_{GB} & \tilde{\mathbf{P}}_{GG} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{FB} \\ \mathbf{x}_{G} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{FB} \\ \mathbf{0}_{G} \end{bmatrix}. \tag{4}$$

$\mathbf{x}_G$ can be eliminated from the bottom line, and then

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{GG}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix} \mathbf{x}_{FB} = \mathbf{y}_{FB}, \tag{5}$$

so that an explicit expression for the preconditioning matrix is obtained

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FB} \\ \mathbf{A}_{BF} & \tilde{\mathbf{P}}_{BB} - \tilde{\mathbf{P}}_{BG}\tilde{\mathbf{P}}_{GG}^{-1}\tilde{\mathbf{P}}_{GB} \end{bmatrix}. \tag{6}$$

A first consequence of this expression is that a lot of eigenvalues of the preconditioned matrix will be 1. Consider the space of all vectors $\mathbf{x}$ such that the B component is null, then

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{x}, \\ \mathbf{P}^{-1}\mathbf{A}\mathbf{x} &= \mathbf{x}, \end{aligned} \tag{7}$$

so that $\mathbf{x}$ is an eigenvector with eigenvalue 1.

2.1 Numerical experiment computing condition numbers

The condition number of matrices for the Poisson problem have been computed with and without preconditioning.

- N_x ranges from 8 to 64.
- The Poisson problem is computed selecting the quadrangles whose center fall outside the body.
- In all cases the domain is the unit square with periodic boundary conditions.
- The bodies considerer are: cylinder of radius 0.2, a vertical strip of width 0.5, and a square of side 0.5.
- Ths condition numbers are computed with Octave.

Note that in all cases the non preconditioned matrix condition number grows as $O(N_x^2)$, whereas with the preconditioning it remains constant.

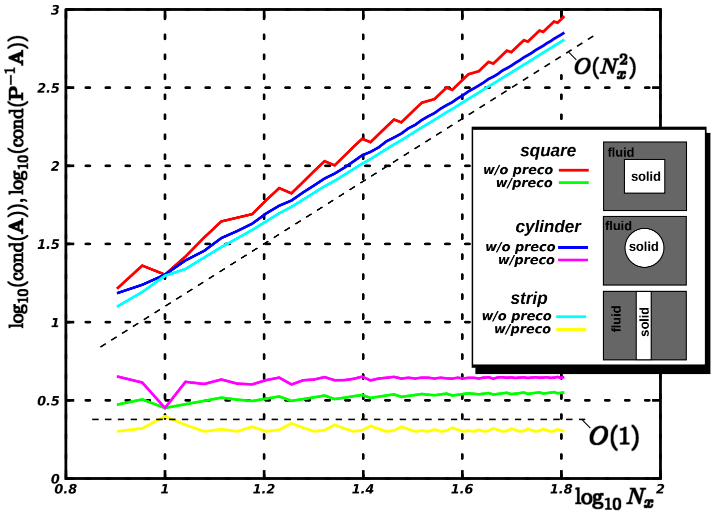

Fig. 2. Condition number of Poisson problem with and without FFT preconditioning

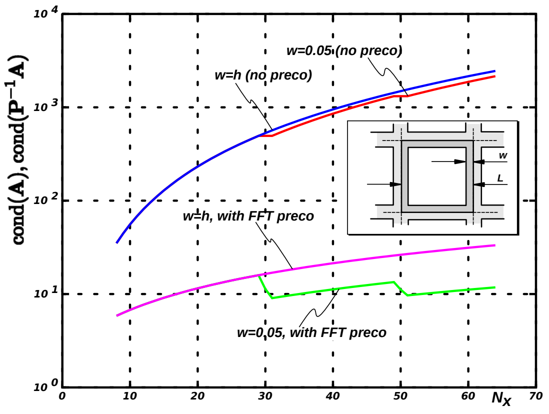

Fig. 3. Condition number for Poisson problem on a square, with and without FFT preconditioning.

2.2 The thin wall case

Consider now the case where the fluid occupies the *interior* of a square $\max(|x - L/2|, |y - L/2|) < L/2 - w$ where w is the width of the wall (see figure 3). In the figure the condition number for the Poisson problem with and without preconditioning are shown. The case where the width of the wall varies so as to have always one element (in fact two elements due to the periodic b.c.'s) separating the squares is considered. On the other hand if a fixed value $w = 0.05$ then the condition number of the preconditioned case is kept bounded.

2.3 Computation of the condition number in terms of the eigenvalues of the Steklov operators

The eigenvalues of the Steklov operators can be computed in closed form for the case of a cylinder in an infinite flow. Recall that the convergence of the AGP scheme is controlled by the condition number of the preconditioned operator

$$\mathrm{cond}(P^{-1}A) = \frac{\max|\,\mathrm{eig}(P^{-1}A)|}{\min|\,\mathrm{eig}(P^{-1}A)|} \tag{8}$$

As all are positive and definite operators, the eigenvalues λ are real and positive, and $0 \le \lambda \le 1$. Also there are a lot of eigenvalues that are unity $\lambda = 1$, they correspond to the space of functions that satisfy the boundary condition $(\partial\phi/\partial n) = 0$ at Γ. However, the Krylov methods iterate only on the space perpendicular to it, and it can be shown that the condition number of the preconditioned Steklov operator must be computed

$$\tilde{\mathcal{S}} = (\mathcal{S}_F + \mathcal{S}_S)^{-1}\mathcal{S}_F. \tag{9}$$

$$\kappa(\tilde{\mathcal{S}}) = \frac{\max[\mathrm{eig}(\tilde{\mathcal{S}})]}{\min[\mathrm{eig}(\tilde{\mathcal{S}})]} \tag{10}$$

For simple geometries like a semi-infinite plane, a strip, and a cylinder the eigenvalues of $\mathcal{S}_F$, $\mathcal{S}_S$ can be explicitely computed. In addition, it turns out that the eigenfunctions are the same, so that the spectral decomposition of the sum $\mathcal{S}_F + \mathcal{S}_S$ and the preconditioned operator are available. Recall that the Steklov $\mathcal{S}_F : V_\Gamma \to V_\Gamma$, operator is defined as $w = \mathcal{S}_F(v)$

$$\begin{aligned} &\Delta\phi = 0, \text{ in } \Omega_F \\ &\phi_\Gamma = v, \text{ and } w = (\partial\phi/\partial n)|_\Gamma \end{aligned} \tag{11}$$

where $V_\Gamma = \{\text{real valued functions on } \Gamma\}$, $\hat{\mathbf{n}}$ is the normal to Γ exterior to Ω_F. The same definition, *mutatis mutandis*, applies to $\mathcal{S}_S$.

The semiplane. The geometry consists on a semiplane

$$\Omega_F = \{\mathbf{x}/x < 0\}, \quad \Omega_S = \{\mathbf{x}/x > 0\}, \quad \Gamma = \{\mathbf{x}/x = 0\} \tag{12}$$

By symmetry of translation in the y direction the eigenfunctions must be plane waves of the form $v = \mathrm{e}^{iky}$ with k real. The solution to the Poisson problem on the fluid and solid are

$$\begin{aligned} &\phi = \mathrm{e}^{iky}\mathrm{e}^{-|k|x}, \ \mathbf{x} \in \Omega_S, \\ &\phi = \mathrm{e}^{iky}\mathrm{e}^{|k|x}, \ \mathbf{x} \in \Omega_F, \end{aligned} \tag{13}$$

and then

$$\mathrm{eig}\{\mathcal{S}_F(v)\} = \mathrm{eig}\{\mathcal{S}_S(v)\} = |k|, \tag{14}$$

so that all the eigenvalues of $\tilde{\mathcal{S}}$ are 0.5, and $\kappa(\tilde{\mathcal{S}}) = 1$.

The cylinder. The domain is $\Omega_S = \{||\mathbf{x}|| = R\}$. By rotational symmetry the eigenfunctions must be

$$\begin{aligned} v_{n,+} &= \cos(n\theta) \\ v_{n,-} &= \sin(n\theta) \end{aligned} \tag{15}$$

where r, θ are polar coordinates at the center of the cylinder. The solution at both domains are

$$\phi_{n,\pm,F} = r^{-n}\begin{Bmatrix}\cos(n\theta)\\ \sin(n\theta)\end{Bmatrix}, \quad \phi_{n,\pm,S} = r^{n}\begin{Bmatrix}\cos(n\theta)\\ \sin(n\theta)\end{Bmatrix}, \tag{16}$$

so that the eigenvalues and eigenfunctions of both operators are the same

$$\lambda(n, \pm, F/S) = \frac{n}{R} \tag{17}$$

and again $\lambda_{n,\tilde{\mathcal{S}}} = 1/2$, $\kappa(\tilde{\mathcal{S}}) = 1$.

The infinite strip The domain is $\Omega_S = \{|x| \leq w/2\}$ where w is the width of the strip. The space V_Γ are pairs of functions on both sides of the strips. By translation invariance in the y direction

$$v = \begin{cases} a\,\mathrm{e}^{iky}, & \text{for } x = -w/2 \\ b\,\mathrm{e}^{iky}, & \text{for } x = +w/2 \end{cases} \tag{18}$$

it can be shown by symmetry $x \to -x$ that the eigenfunctions are the symmetric ($a = b$) and antisymmetric ($a = -b$) combinations, so that

$$v_{k,\pm} = \begin{cases} \pm\mathrm{e}^{iky}, & \text{for } x = -w/2, \\ \mathrm{e}^{iky}, & \text{for } x = +w/2. \end{cases} \tag{19}$$

The corresponding solution for the symmetric modes at $\Omega_{F,S}$ are

$$\phi_{k,+} = \begin{cases} \mathrm{e}^{iky+|k|(w/2-x)}, & \text{for } |x| \geq w/2, \\ \dfrac{\cosh(kx)}{\cosh(kw/2)}\mathrm{e}^{iky}, & \text{for } |x| \leq w/2, \end{cases} \tag{20}$$

and the corresponding eigenvalues

$$\begin{aligned} \lambda(k, +, F) &= |k|, \\ \lambda(k, +, S) &= k\tanh(kw/2). \end{aligned} \tag{21}$$

And for the antisymmetric eigenfunctions,

$$\phi_{k,-} = \begin{cases} \operatorname{sign}(x)\mathrm{e}^{iky+|k|(w/2-x)}, & \text{for } |x| \geq w/2, \\ \dfrac{\sinh(kx)}{\sinh(kw/2)}\mathrm{e}^{iky}, & \text{for } |x| \leq w/2, \end{cases} \tag{22}$$

and the corresponding eigenvalues

$$\begin{aligned} \lambda(k, -, F) &= |k|, \\ \lambda(k, -, S) &= k\coth(kw/2). \end{aligned} \tag{23}$$

Both the symmetric and antisymmetric eigenvalues van be seen in figure 4. Note that the eigenvalues of the fluid operator $\lambda(k, \pm)$ are the same for the symmetric and antisymmetric case and independent of the strip width w, since the fluid domain is completely decoupled by the strip, and then the eigenvalues of the Steklov operator $\mathcal{S}_F$ are the same as those of those of each semiplane $x > w/2$ and $x < w/2$.

On the other hand, with respect to the eigenvalues of the Steklov operator of the strip (solid domain) $\mathcal{S}_S$, note that both symmetric and antisymmetric eigenvalues behave like $\sim |k|$ for $kw \to \infty$. This is because for kw large means that the wavelength of the eigenfunction is much smaller than the width of the strip and then again, the behavior of the eigenvalues is the same as for a infinite semiplane.

However, when kw is small the behavior of the symmetric and antisymmetric branches is very different. Remember that, as per definition of the Steklov operator, given an eigenfunction u it must be imposed as a Dirichlet boundary condition on the interface

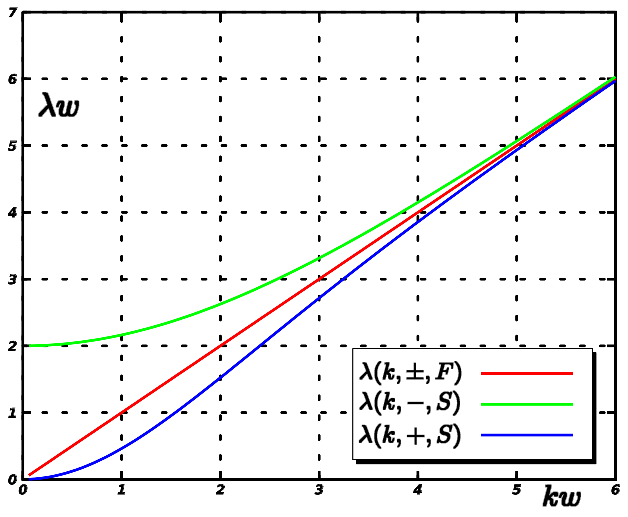

Fig. 4. Eigenvalues of Steklov operators for the solid strip case.

Γ, solve the Laplace equation for the field ϕ in the corresponding domain and compute the flux $v = (\partial\phi/\partial n)$. As u is an eigenfunction, v should be proportional to u and the proportionality constant is the eigenvalue λ. First consider the symmetric branch. If a sinusoidal value is imposed on the left boundary $x = -w/2$ (see figure 6) then for the symmetric mode the same Dirichlet boundary condition must be imposed on the other boundary $x = +w/2$. As a result, facing points inside the strip like A, A' or B, B' have equal values of temperature ϕ imposed and then the heat flow is very low, which means a small eigenvalue. On the other hand, for the antisymmetric mode, opposing points have the same absolute temperature but of opposed sign, and the heat flow is very high (the red arrows in the figure). This explains why for low wavenumber k the eigenvalues of the symmetric mode are smaller, with a behavior $\lambda \propto k^2$ for $k \to 0$. For the antisymmetric mode for low wavenumber the eigenvalue is larger with a behavior $\lambda w \to 2$ for $kw \to 0$.

This last limit is simple to understand. Effectively, for very small wavenumber conduction in the y direction can be neglected, and so for an eigenfunction $u = \pm\cos ky$ at $x = \pm w/2$ the solution is

$$\phi = \frac{2x}{w}\cos ky, \tag{24}$$

so that

$$v = (\partial\phi/\partial n)|_{x=w/2} = \frac{2}{w}\cos ky = \frac{2}{w}u. \tag{25}$$

So that, $\lambda w = 2$.

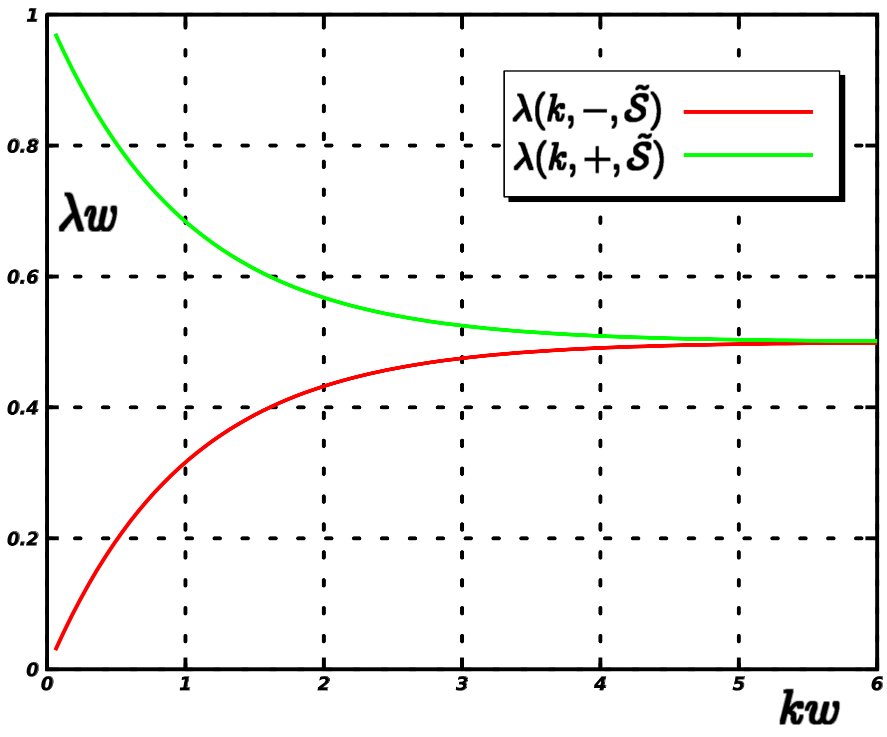

Fig. 5. Eigenvalues of preconditioned Steklov operator for the symmetric and antisymmetric branches.

The eigenvalues of the preconditioned Steklov operators are plot in figure 5. They are given by the expressions

$$\begin{aligned} \lambda(k,+,\tilde{\mathcal{S}}) &= \frac{|k|}{|k| + k\tanh(kw/2)}, \\ \lambda(k,-,\tilde{\mathcal{S}}) &= \frac{|k|}{|k| + k\coth(kw/2)}. \end{aligned} \tag{26}$$

Note that for both symmetric and antisymmetric mode the eigenvalues g to $1/2$ for $kw \to \infty$, as for an infinite semiplane. On the other hand, for small kw the eigenvalues of the symmetric mode are larger than $1/2$, and those for the antisymmetric modes are smaller. So, if the eigenvalues for the symmetric modes are considered the condition of the preconditioned Steklov operator is 2, whereas if we consider the antisymmetric modes the condition number tends to ∞ since the smallest eigenvalue tends to 0 for $kw \to 0$.

This is expected, since the FFT preconditioning is based on solving the Poisson equation on the whole domain, fluid and solid, instead in solving only on the fluid. Thus, this preconditioning is good whenever the fluxes in the solid domain are small. But this is exactly the case for the symmetric modes, and the opposite happens for the antisymmetric modes.

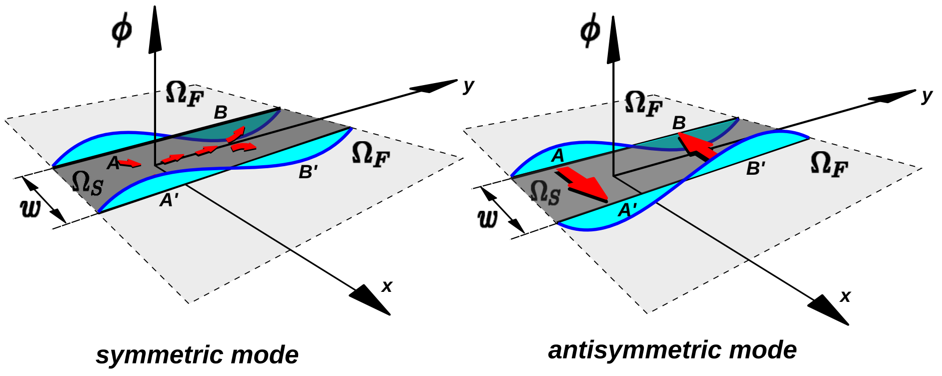

Fig. 6. Explanation of the behavior of the Steklov eigenvalues for large wavelengths.

Estimation of the condition number for thin walls The analysis of the infinite strip shows that a situation where the FFT preconditioning is deteriorated is when there are thin walls in the solid geometry, since in that case the modes that are antisymmetric about the axis of the solid produce large heat fluxes in the solid, and this is an indication of bad performance of the preconditioning. So for elongated solid geometries with, say, a typical length of L and a typical width of $w \ll L$ an estimate of the condition number of the preconditioned operator can be obtained by taking L as the maximal wavelength and the estimate gives a condition number of

$$\kappa(\tilde{\mathcal{S}}) \sim \frac{|k_{\min}| + k_{\min} \coth(k_{\min} w/2)}{|k_{\min}|} \tag{27}$$

where $k_{\min} = 2\pi/L$. By algebraic manipulation this can be simplified to

$$\kappa(\tilde{\mathcal{S}}) \sim 1 + \coth(\frac{\pi w}{L}) \sim \frac{L}{\pi w}, \tag{28}$$

i.e. the condition number is proportional to the aspect ratio of the solid domain.

3 CUDA implementation

A basic description of the CUDA implementation will be given here. Complete details can be found elsewhere [4, 5].

Code (1) shows the pseudocode for the complete set of steps required under a fractional-step method to solve checkerboard problems on pressure-velocity decoupling under discrete schemas as finite differences or volume methods.

Algorithm 1 - Pseudocode used to explain briefly the steps requiered by our problem.

```
for i=1:TimeSteps {
  1- Update solid position.
  2- Impose solid velocities on current velocity field.
  3- Solve momentum equations.
  4- Time integration (Adams-Bashfort)
  5- Impose solid velocities on current velocity field.
  6- Compute velocity divergence.
  7- CG+FFT solver.
    7.1- Fluid+solid FFT solver.
    7.2- Solving Poisson equation for pressure (↩
         contributions accounted only by solid-free nodes).
  8- Compute pressure gradients (contributions accounted ↩
     only by solid-free nodes).
  9- Update solid-free nodes using pressure gradients.
}
```

Some comments about the algorithm follows:

- Step 1 applies only in the case of moving bodies. If the bodies are at rest this step is irrelevant.
- Step 2 consists in imposing the velocities of the solid to those velocity cells that fall inside the given body.
- In step 7.2 only the contributions of cells not connected to the solid are assembled.
- In steps 8-9 pressure gradients are obtained and used to update velocity nodes whose pressure neighbours are not in the solid.

The results obtained by the algorithm are shown on Table (1). The GPGPU used was a NVIDIA Tesla C2050 under a CPU intel i7 950. CG iterations were limited on 3, where an absolute error tolerance of $10^{-2} \sim 10^{-3}$ was reached.

	Simple [segs/MCels]	Double [segs/MCels]
32x32x32	0.17	0.20
64x64x64	0.043	0.062
128x128x128	0.026	0.044

Table 1. Performance obtained per time step using NVIDIA Tesla C2050. CG iterations: 3.

The most consuming steps are those on solving momentum equations and the Poisson step.

In this work the NVIDIA cuFFT library [16, 17] has been used to perform FFT's, Thrust and CUSP API's to manage vector and linear algebra operations, and VTK for visualization the results obtained.

4 Numerical experiments

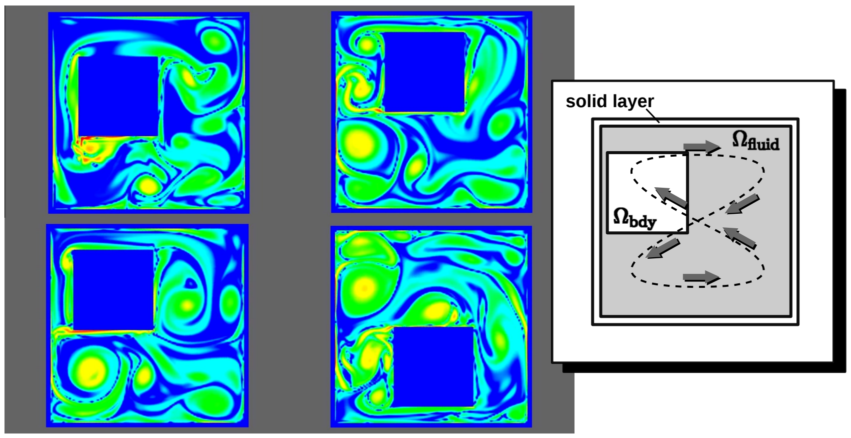

Fig. 7. Colormap of $\log_{10}(|\omega|)$ for a square of side $L_s = 0.4$[m] moving in a square domain of side $L = 1$[m]. The square moves forming a Lissajous 8-shaped curve.

Numerical simulations of several flows involving moving bodies are shown in figures 7-11. In all cases (except for the case of the example in section §4) the flows represent a body moving inside a square or cubic cavity of length side 1[m]. In order to circumvent the restriction of periodic boundaries intrinsic to the FFT solver, a thin layer (2.5% of the square or cubic domain side length) is defined as a fixed body. In all cases the color corresponds to $\log_{10}(|\omega|)$, i.e. the absolute magnitude of the vorticity vector $\omega = \nabla \times \mathbf{u}$ in logarithmic scale. This quantity helps in the visualization of boundary layers, since the magnitude of vorticity has variations of several orders of magnitude in flows with boundary layers at high Reynolds numbers. In 2D cases the mesh was 128×128 and in 3D cases $128 \times 128 \times 128$. In all cases the side of the domain (square in 2D, cube in 3D) was $L = 1$[m] and the kinematic viscosity was $\nu = 6.33 \times 10^{-5}[\mathrm{m}^2/\mathrm{s}]$.

Square moving in curved trajectory. The body is a square of side $L_s = 0.4$[m], and the center of the body (x_c, y_c) describes an 8-shaped Lissajous curve, described by

$$\begin{aligned} &x_c = \frac{L}{2} + A\cos(2\omega t), y_c = \frac{L}{2} + A\cos(\frac{\pi}{2} + \omega t), \\ &\omega = 1[\mathrm{s}^{-1}],\ A = 0.2[\mathrm{m}] \end{aligned} \tag{29}$$

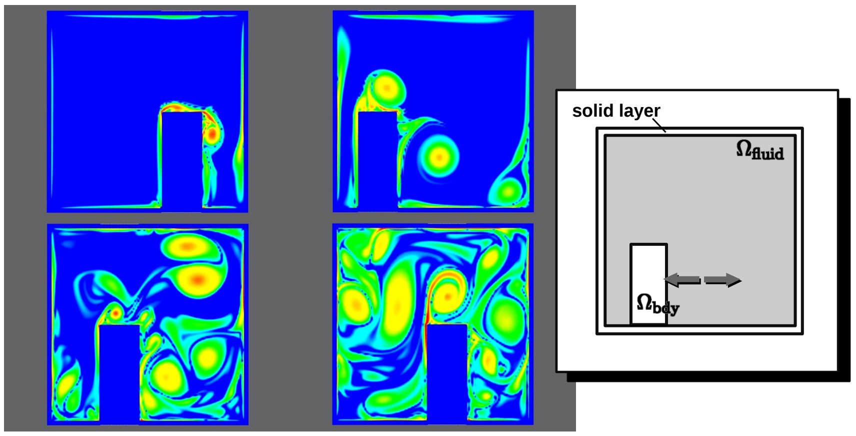

Fig. 8. Colormap of $\log_{10}(|\omega|)$ for a rectangle sliding on the bottom of the domain.

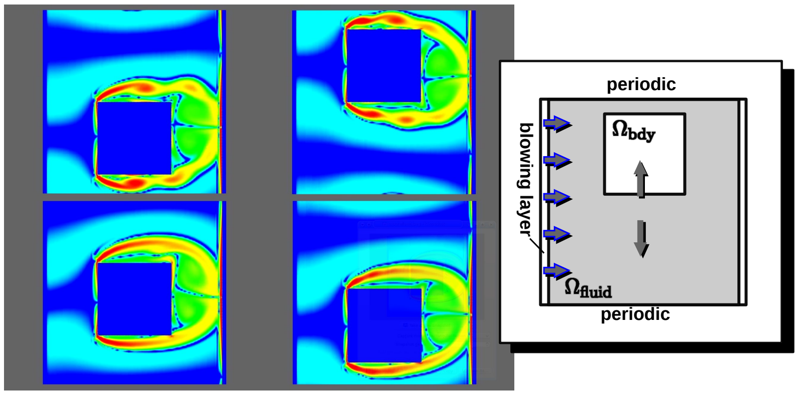

Fig. 9. Colormap of $\log_{10}(|\omega|)$ for a square body performing harmonic motion in the vertical direction with a cross flow in the horizontal direction.

As the body displaces fluid high levels of vorticity can be observed at the vertices. As the simulation progresses large vortices remain rotating in the fluid with long filamentary vorticity layers that are a characteristic 2D feature (they are unstable in 3D).

Moving rectangular obstacle. The body is a rectangle of height $H = 0.5[\mathrm{m}]$ and width $W = 0.2[\mathrm{m}]$. An harmonic horizontal displacement as follows

$$\begin{aligned} x_c &= (L/2) + A\cos(\omega t), \\ \omega &= 1[\mathrm{s}^{-1}],\ A = 0.3[\mathrm{m}], \end{aligned} \tag{30}$$

is imposed. As the body displaces fluid a large concentration of vorticity is observed in the upper corner of the body, with characteristic trailing filamentary vortex layers that detach from the corners.

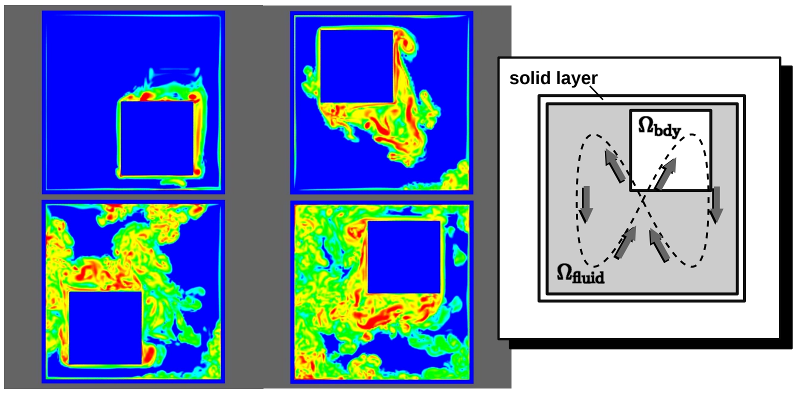

Fig. 10. Colormap of $\log_{10}(|\omega|)$ for a cube moving in a Lissajous 8-shaped curve.

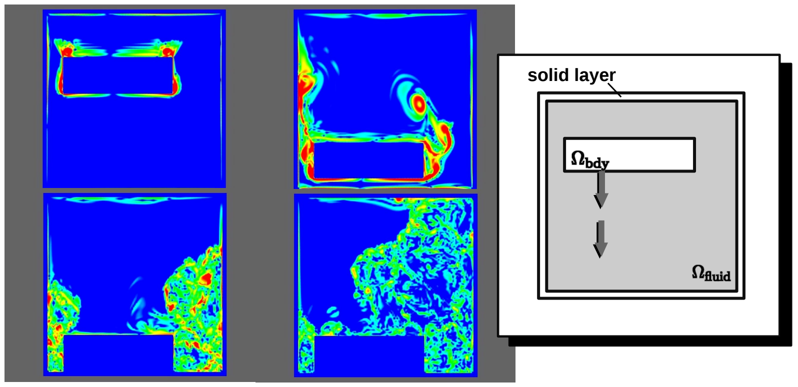

Fig. 11. Colormap of $\log_{10}(|\omega|)$ for a falling block.

Square moving vertically with mean horizontal flow. In this example the exterior boundary of the computational domain is not at rest, but rather it is intended to generate a mean flow that impinges on the body. This freestream flow is obtained with a layer of width 0.025[m] at the left and right sides were a positive x velocity of $u = 1[\mathrm{m/s}]$ is imposed. Periodic boundary conditions are imposed in the vertical y direction. The body is a square of side $L_s = 0.4[\mathrm{m}]$, the center of the body (x_c, y_c) is centered in the x direction and experiences an harmonic vertical movement

$$\begin{aligned} y_c &= (L/2) + A\cos(\omega t), \\ \omega &= 0.5[\mathrm{s}^{-1}],\ A = 0.2[\mathrm{m}]. \end{aligned} \tag{31}$$

An accelerating boundary layer is formed at the left side facing the fluid stream. The boundary layer accelerate towards the corners and detach there. If the vertical movement were at a constant velocity then the flow would be equivalent to a fixed body with an impinging stream at an angle of attack. A notable feature of the flow is that when the body reaches the extreme positions in the y direction the vortex layers become unstable and start shedding vortices, whereas when the body is moving the vortex layer stabilizes.

Moving cube This is a 3D case. The center (x_c, y_c, z_c) of a cube of side $L_s = 0.4[\mathrm{m}]$ is describing a Lissajous 8-shaped figure in the $z = 0.66[\mathrm{m}]$ plane, as follows

$$\begin{aligned} x_c &= L/2 + A\cos(\omega t), \\ y_c &= L/2 + A\cos(\frac{\pi}{2} + 2\omega t), \\ z_c &= 0.66[\mathrm{m}], \\ \omega &= 2[\mathrm{s}^{-1}], A = 0.4[\mathrm{m}] \end{aligned} \tag{32}$$

This is similar to the case §4 but 3D. The large filamentary vortex layers are no more present, but instead there is a large amount of small eddies characteristic of a 3D flow.

Falling block The body is a parallelepiped block of dimensions $L_x = L_z = 0.6[\mathrm{m}]$, $L_y = 0.2[\mathrm{m}]$. The center of the body is initially at $(x_c, y_c, z_c) = (0.4125, 0.7, 0.5)[\mathrm{m}]$ and starts falling vertically with a velocity of $1[\mathrm{m/s}]$. As the body falls it displaces a large quantity of fluid that forms a turbulent region expanding from both sides of the block.

5 Conclusions

A new method called Accelerated Global Preconditioning for solving the incompressible Navier-Stokes equations with moving bodies was presented. The algorithm is based on a pressure segregated, staggered grid, Finite Volume formulation and uses an FFT solver for preconditioning the CG solution of the Poisson problem. Theoretical estimates of the condition number of the preconditioned Poisson problem are given, and several numerical examples are presented validating these estimates. The algorithm is specially suited for implementation on GPU hardware. The condition number of the preconditioned Poisson equation does not degrade with refinement. The algorithm allows computing 3D problems in real time on moderately large meshes for many problems of practical interest in the area of Computational Fluid Dynamics.

6 Acknowledgment

This work has received financial support from

- Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, PIP 5271/05),
- Universidad Nacional del Litoral (UNL, Argentina, grant CAI+D 2009-65/334),
- Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1506/2006, PICT-1141/2007, PICT-0270/2008), and

– European Research Council (ERC) Advanced Grant, Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG).

The authors made extensive use of *Free Software* as GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software as VTK among many others. In addition, many ideas from these packages have been inspiring to them.

Bibliography

[1] Adams, S., Payne, J., Boppana, R.: Finite difference time domain (FDTD) simulations using graphics processors. HPCMP Users Group Conference 0, 334–338 (2007)

[2] Bell, N., Garland, M.: Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. pp. 1–11. ACM, New York, NY, USA (2009)

[3] Corrigan, A., Camelli, F.F., Löhner, R., Wallin, J.: Running unstructured grid-based CFD solvers on modern graphics hardware. International Journal for Numerical Methods in Fluids (2010), (in press)

[4] Costarelli, S.: Resolución de las ecuaciones de navier-stokes utilizando cuda. Tech. rep., Universidad Nacional del Litoral (2011), http://www.cimec.org.ar/ojs/index.php/cimec-repo/article/view/3735

[5] Costarelli, S., Paz, R., Dalcin, L., Storti, M.: Resolución de las ecuaciones de navier-stokes utilizando cuda. In: Muller, O., Signorelli, J., Storti, M. (eds.) Mecánica Computacional. vol. XXX, pp. 2979–3008. AMCA (2011), http://www.cimec.org.ar/ojs/index.php/mc/article/view/3965

[6] Crane, K., Llamas, I., Tariq, S.: Chapter 30 - Real-Time Simulation and Rendering of 3D Fluids (2008)

[7] Elcott, S., Tong, Y., Kanso, E., Schröder, P., Desbrun, M.: Stable, circulation-preserving, simplicial fluids. In: SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses. pp. 1–11. ACM, New York, NY, USA (2008)

[8] Elsen, E., LeGresley, P., Darve, E.: Large calculation of the flow over a hypersonic vehicle using a GPU. J. Comput. Phys. 227(24), 10148–10161 (2008)

[9] Goddeke, D., Strzodka, R., Mohd-Yusof, J., McCormick, P., Wobker, H., Becker, C., Turek, S.: Using GPU's to improve multigrid solver performance on a cluster. Int. J. Comput. Sci. Eng. 4(1), 36–55 (2008)

[10] Irving, G., Guendelman, E., Losasso, F., Fedkiw, R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. ACM Trans. Graph. 25(3), 805–811 (2006)

[11] Klöckner, A., Warburton, T., Bridge, J., Hesthaven, J.: Nodal discontinuous galerkin methods on graphics processors. Journal of Computational Physics 228(21), 7863–7882 (2009)

[12] Lastra, M., Mantas, J.M., Ure na, C., Castro, M.J., García-Rodríguez, J.A.: Simulation of shallow-water systems using graphics processing units. Math. Comput. Simul. 80(3), 598–618 (2009)

[13] Molemaker, J., Cohen, J.M., Patel, S., Noh, J.: Low viscosity flow simulations for animation. In: SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 9–18. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2008)

[14] Mossaiby, F., Rossi, R., Dadvand, P., Idelsohn, S.: Opencl-based implementation of an unstructured edge-based finite element convection-diffusion solver on graphics hardware. International Journal for Numerical Methods in Engineering 89, 1635 1651 (2012)

[15] Mullen, P., Crane, K., Pavlov, D., Tong, Y., Desbrun, M.: Energy-preserving integrators for fluid animation. In: SIGGRAPH '09: ACM SIGGRAPH 2009 papers. pp. 1–8. ACM, New York, NY, USA (2009)

[16] Nvidia, C.: Compute unified device architecture (CUDA) (2010), `http://developer.nvidia.com/category/zone/cuda-zone`

[17] Nvidia, C.: CUFFT library (2010), `http://developer.nvidia.com/cufft`

[18] P.Rinaldi, Bauza, C.G., Vénere, M., Clausse, A.: Paralelización de autómatas celulares de aguas superficiales sobre placas gráficas. In: Cardona, A., Storti, M., Zuppa, C. (eds.) Mecánica Computacional Vol. XXVII. vol. XXVII, pp. 2943–2957 (2008)

[19] Ryoo, S., Rodrigues, C.I., Baghsorkhi, S.S., Stone, S.S., Kirk, D.B., Hwu, W.m.W.: Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In: PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. pp. 73–82. ACM, New York, NY, USA (2008)

[20] Thibault, J.C., Senocak, I.: CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. In: AIAA (ed.) 47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition (Disc 1) (2009)

[21] Wang, X., Wang, C., Zhang, L.: Semi-implicit formulation of the immersed finite element method. Computational Mechanics 49, 421–430 (2012)

[22] Wu, E., Liu, Y., Liu, X.: An improved study of real-time fluid simulation on GPU: Research articles. Comput. Animat. Virtual Worlds 15(3-4), 139–146 (2004)